

# NCOMMAS Collaboration: Rice Component

(part 2 of a longer presentation)

## A.K.A. – Woodward in a Box

Rob Fowler

John Mellor-Crummey

Guohua Jin

Dick Hanson

Ken Kennedy

Center for High Performance Software  
Rice University



# Overview

1. Algorithmic Issues
  - A. Coding to not prevent optimization, e.g. use modern calling conventions, memory allocation.
  - B. Start with best algorithms possible.
2. Performance Analysis
  - A. Apply Rice HPC Tools suite.
3. Tools to improve spatial and temporal reuse of data.
  - A. Loop fusion, eliminate/reduce temporaries
  - B. Time skewing on-node (SC2001 paper)
  - C. Transformations to “coarsen” parallel computations.

## 3A. Source-to-source Transformations

### Current capabilities --

- + Aggressive loop fusion at multiple levels
  - + Apply code motion to enable fusion
  - + Skew spatial loops to enable fusion and blocking
  - + Generate code with a guard-free core

### Soon --

- + Reduce size of temporaries used in fused and blocked code
- + Skew spatial and time loops to improve temporal locality
- + Block and unroll loops for register reuses

# Example: NCOMMAS advection kernel

```
thirddtbydy = dt / (3. * dy); halfdtbydy = 1.5 * thirddtbydy
do j = 4-nbdy,ny+nbdy-2, i = 4-nbdy,nx+nbdy-3, k = 4-nbdy,nz+nbdy-3
    ab(k,i,j) = (f60 * (a(k,i,j-1) + a(k,i,j)) + f61 * (a(k,i,j-2) + a(k,i,j+1))
                  + f62 * (a(k,i,j-3) + a(k,i,j+2))) * thirddtbydy * uyb(k,i,j)
    thirddtbydx = dt / (3. * dx); halfdtbydx = 1.5 * thirddtbydx
! ... j,i,k loops for computing al array
    thirddtbydz = dt / (3. * dz); halfdtbydz = 1.5 * thirddtbydz
! ... j,i,k loops for computing af array
    do j = 4-nbdy,ny+nbdy-3, i = 4-nbdy,nx+nbdy-3, k = 4-nbdy,nz+nbdy-3
        athird(k,i,j) = a(k,i,j) + (al(k,i+1,j) - al(k,i,j)) + (ab(k,i,j+1) - ab(k,i,j)) + (af(k+1,i,j) - af(k,i,j))
    do j = 7-nbdy,ny+nbdy-5, i = 7-nbdy,nx+nbdy-6, 7-nbdy,nz+nbdy-6
        abthird(k,i,j) = (f60 * (athird(k,i,j-1) + athird(k,i,j)) + f61 * (athird(k,i,j-2) + athird(k,i,j+1))
                           + f62 * (athird(k,i,j-3) + athird(k,i,j+2))) * halfdtbydx * uybthird(k,i,j)
! ... j,i,k loops for computing althird and afthird array
    do j = 7-nbdy,ny+nbdy-6, i = 7-nbdy,nx+nbdy-6, k = 7-nbdy,nz+nbdy-6
        ahalf(k,i,j) = a(k,i,j) + (althird(k,i+1,j) - althird(k,i,j)) + (abthird(k,i,j+1) - abthird(k,i,j))
&           + (afthird(k+1,i,j) - afthird(k,i,j))
```

# Automatic Fusion of Outermost Loops

```
thirddtbydx_dtmp = dt / (3. * dx); halfdtbydx_dtmp = 1.5 * thirddtbydx_dtmp;
thirddtbydz_dtmp = dt / (3. * dz); thirddtbydy = dt / (3. * dy)
if (4 * nbdy .ge. - nx - nz + 13 .and. 2 * nbdy .ge. - nx + 6 .and. 2 * nbdy .ge. - nz + 6) then
    do j = -nbdy+4, min(ny+3*nbdy+nx-9, ny+3*nbdy+nz-9, ny+nbdy-2)
        if (2*nbdy .ge. -nz+7) then
            do i = 4-nbdy,nx+nbdy-3, k = 4-nbdy,nz+nbdy-3
                ab(k,i,j) = (f60 * (a(k,i,j-1) + a(k,i,j)) + f61 * (a(k,i,j-2) + a(k,i,j+1))
                               + f62 * (a(k,i,j-3) + a(k,i,j+2))) * thirddtbydy * uyb(k,i,j)
        ! ... i,k loops for computing al, af arrays
        if (2*nbdy .ge. -nz+7 .and. j .ge. -nbdy+5) then
            do i = 4-nbdy,nx+nbdy-3, k = 4-nbdy,nz+nbdy-3
                athird(k,i,j-1) = a(k,i,j-1) + (al(k,i+1,j-1) - al(k,i,j-1))
                               + (ab(k,i,j) - ab(k,i,j-1)) + (af(k+1,i,j-1) - af(k,i,j-1))
            if (2*nbdy .ge. -nz+13 .and. j .ge. -nbdy+10) then
                do i = 7-nbdy,nx+nbdy-6, 7-nbdy,nz+nbdy-6
                    abthird(k,i,j-3) = (f60 * (athird(k,i,j-4) + athird(k,i,j-3)) + f61 * (athird(k,i,j-5)
                               + athird(k,i,j-2)) + f62 * (athird(k,i,j-6)
                               + athird(k,i,j-1))) * halfdtbydx_dtmp * uybthird(k,i,j-3)
        ! ... i,k loops for computing althird and afthrid array
        if (2*nbdy .ge. -nz+13 .and. j .ge. -nbdy+11) then
            do i = 7-nbdy,nx+nbdy-6, k = 7-nbdy,nz+nbdy-6
                ahalf(k,i,j-4) = a(k,i,j-4) + (althird(k,i+1,j-4) - althird(k,i,j-4))
                               + (abthird(k,i,j-3) - abthird(k,i,j-4)) + (afthrid(k+1,i,j-4) - afthrid(k,i,j-4))
```

# Reduced Temporaries After Outer Fusion (handcoded)

```
thirddtbydx_dtmp = dt / (3. * dx); halfdtbydx_dtmp = 1.5 * thirddtbydx_dtmp;
thirddtbydz_dtmp = dt / (3. * dz); thirddtbydy = dt / (3. * dy)
if (4 * nbdy .ge. - nx - nz + 13 .and. 2 * nbdy .ge. - nx + 6 .and. 2 * nbdy .ge. - nz + 6) then
do j = -nbdy+4, min(ny+3*nbdy+nx-9, ny+3*nbdy+nz-9, ny+nbdy-2)
    if (2*nbdy .ge. -nz+7) then
        do i = 4-nbdy,nx+nbdy-3, k = 4-nbdy,nz+nbdy-3
            ab(k,i,iand(j,1)) = (f60 * (a(k,i,j-1) + a(k,i,j)) + f61 * (a(k,i,j-2) + a(k,i,j+1))
                + f62 * (a(k,i,j-3) + a(k,i,j+2))) * thirddtbydy * uyb(k,i,j)
!       ... i,k loops for computing al, af arrays
        if (2*nbdy .ge. -nz+7 .and. j .ge. -nbdy+5) then
            do i = 4-nbdy,nx+nbdy-3, k = 4-nbdy,nz+nbdy-3
                athird(k,i,iand(j-1,7)) = a(k,i,j-1) + (al(k,i+1) - al(k,i))
                    + (ab(k,i,iand(j,1)) - ab(k,i,iand(j-1,1))) + (af(k+1,i) - af(k,i))
            if (2*nbdy .ge. -nz+13 .and. j .ge. -nbdy+10) then
                do i = 7-nbdy,nx+nbdy-6, 7-nbdy,nz+nbdy-6
                    abthird(k,i,iand(j-3,1)) = (f60 * (athird(k,i,iand(j-4,7)) + athird(k,i,iand(j-3,7)))
                        + f61 * (athird(k,i,iand(j-5,7)) + athird(k,i,iand(j-2,7)))
                        + f62 * (athird(k,i,iand(j-6,7))
                        + athird(k,i,iand(j-1,7)))) * halfdtbydx_dtmp * uybthird(k,i,j-3)
!               ... i,k loops for computing althird and afthird array
            if (2*nbdy .ge. - nz+13 .and. j .ge. -nbdy+11) then
                do i = 7-nbdy,nx+nbdy-6, k = 7-nbdy,nz+nbdy-6
                    ahalf(k,i,j-4) = a(k,i,j-4) + (althird(k,i+1) - althird(k,i)) + (abthird(k,i,iand(j-3,1))
                        - abthird(k,i,iand(j-4,1))) + (afthird(k+1,i) - afthird(k,i))
```

# Guard-free Core after Multilevel Fusion

! ... Code for top portion

! ... Code for left portion

! ... Code for front portion

```
if (nx .ge. (-2 * nbdy) + 15 .and. 2 * nbdy .ge. (-nz) + 15) then
    do j = -nbdy + 11, nbdy + ny - 5; i = -nbdy+9, nx + nbdy - 6; k = -nbdy+9, nbdy+nz- 6
        ab(k, i, j) = (f60 * (a(k, i, -1 + j) + a(k, i, j)) + f61 * (a(k, i, -2 + j) + a(k, i, 1 + j))
                        + f62 * (a(k, i, -3 + j) + a(k, i, 2 + j))) * thirddtbydy * uyb(k, i, j)
```

! ... Compute al and af arrays

```
athird(k, i, -1 + j) = a(k, i, -1 + j) + (al(k, 1 + i, -1 + j) - al(k, i, -1 + j))
                        + (ab(k, i, j) - ab(k, i, -1 + j)) + (af(1 + k, i, -1 + j) - af(k, i, -1 + j))
abthird(k, i, -3 + j) = (f60 * (athird(k, i, -4 + j) + athird(k, i, -3 + j))
                           + f61 * (athird(k, i, -5 + j) + athird(k, i, -2 + j)) + f62 * (athird(k, i, -6 + j)
                           + athird(k, i, -1 + j))) * halfdtbydx_dttmp * uybthird(k, i, -3 + j)
```

! ... Compute althird and afthrid arrays

```
ahalf(k, i, -4 + j) = a(k, i, -4 + j) + (althird(k, 1 + i, -4 + j) - althird(k, i, -4 + j))
                        + (abthird(k, i, -3 + j) - abthird(k, i, -4 + j))
                        + (afthrid(1 + k, i, -4 + j) - afthrid(k, i, -4 + j))
```

! ... Code for back portion

! ... Code for right wing

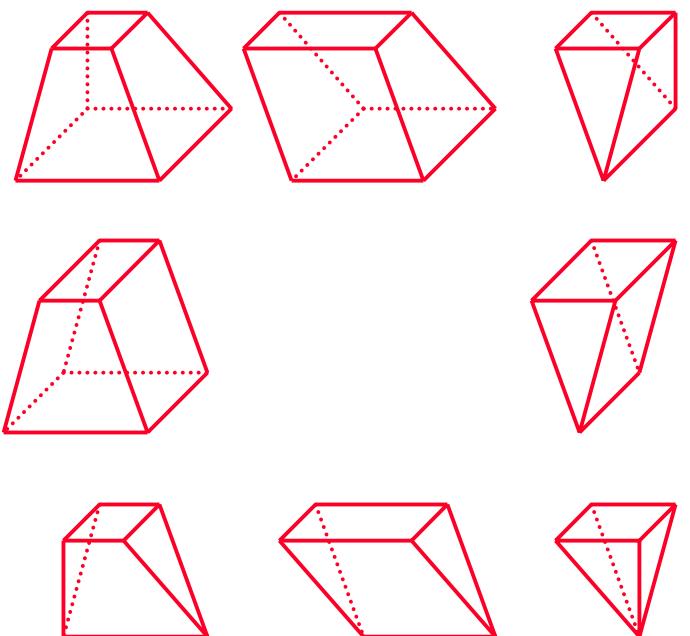
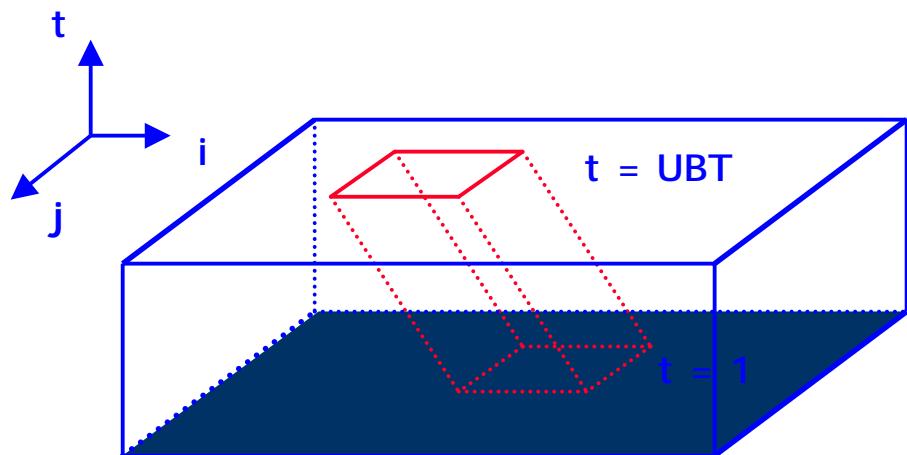
! ... Code for bottom plane

# Prismatic Time Skewing (SC2001)

Iterative Computations, e.g. discrete time simulation

- Problem
  - poor temporal cache reuse at all levels for large data size
- Approach: Exploit reuse across “iterations”
  - divide spatial domain into small blocks
  - advance a block multiple time steps before going to next block
  - use enhanced loop skewing to handle more complex cases
  - combines nicely with other approaches
    - » (recursive) blocking for enhancing cache temporal reuse
    - » unroll and jam, scalar replacement, ...

# 2D Prismatic Time Skewing w/ Blocking



# 2D Prismatic Time Skewing w/ Blocking

```
DO t= 1, Ut  
  DO i = 1,Ui  
    DO j = 1,Uj  
      a(j,i) = c1 * (a(j+1,i) + a(j-1,i) +  
                      a(j,i+1) + a(j,i-1) + c2 * a(j,i))
```



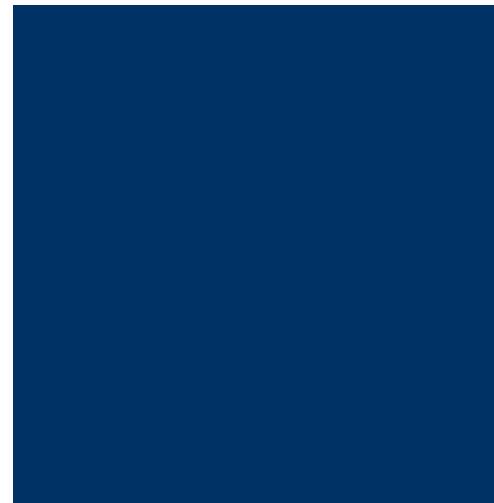
```
Subroutine prism(lbi,lbj,...)
```

```
...
```

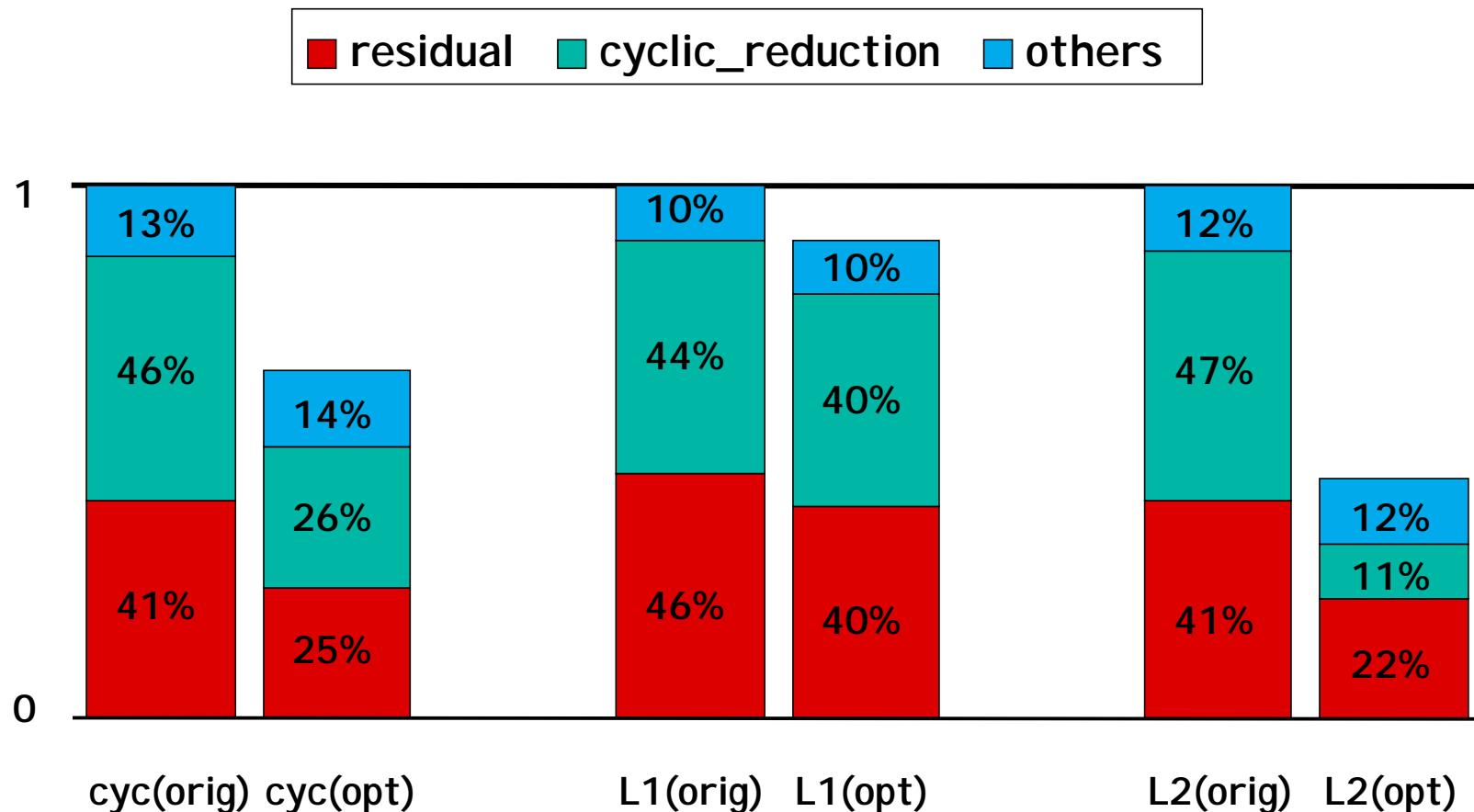
```
  DO t= 1, Ut  
    DO i = max(lbi-t+1,1), min(lbi+Bi-t,Ui)  
      DO j = max(lbj-t+1,1), min(lbj+Bj-t,Uj)  
        a(j,i) = c1 * (a(j+1,i) + a(j-1,i) +  
                        a(j,i+1) + a(j,i-1) + c2 * a(j,i))
```



T=1  
T=2  
T=3  
T=4  
T=5



# Performance Evaluation (Smg98)



# Balanced Backfill Time Skewing in 1D

